EAST Search History

Ref #	Hits	Search Query	DBs	Default Operator	Plurals	Time Stamp
L1	914	annotat$4 with modif$5	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/05/09 13:35
L2	1292	annotat$4 with (modif$5 modificat$5)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/05/09 13:36
L3	300	annotat$4 with (modif$5 modificat$5) and annotat$4 with program	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/05/09 13:37
L4	8	annotat$4 with (modif$5 modificat$5) and annotat$4 with program and warning with (annotat$4 modif$5 modificat$5)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/05/09 13:43
L5	12	map$4 with warning with (annotat$4 modif$5 modificat$5)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/05/09 13:44
L6	3	map$4 with warning with (annotat$4 with (modif$5 modificat$5))	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/05/09 13:45
L7	3	map$4 with warning with (annotat$4) and annotat$4 with program	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/05/09 13:45
L8	18	warning with (annotat$4) and annotat$4 with program	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/05/09 13:48

EAST Search History

L9	75	717/130-131.ccls. and annotat$4	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/05/09 13:58
L10	641	717/130-131.ccls.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/05/09 13:59
L11	344	717/130.ccls.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/05/09 13:59
L12	21	debug with annotat$4	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/05/09 14:00

Results for "(((annotation<in>metadata) <and> (program<in>metadata))<and> (modify&..." e-mail
Your search matched 3 of 1348795 documents.
A maximum of 100 results are displayed, 25 to a page, sorted by Relevance in Descending order.

» Search Options

View Session History

New Search

Modify Search

(((annotation<in>metadata) <and> (program<in>metadata))<and> (modify<in>me Search

Check to search only within this results set

Display Format: Citation Citation & Abstract

» Key

IEEE JNL	IEEE Journal or Magazine
IEE JNL	IEE Journal or Magazine
IEEE CNF	IEEE Conference Proceeding
IEE CNF	IEE Conference Proceeding
IEEE STD	IEEE Standard

View selected items Select All Deselect All

1. **MicroScope: a knowledge-based programming environment**
Ambras, J.; O'Day, V.;
Software, IEEE
Volume 5, Issue 3, May 1988 Page(s):50 - 58
Digital Object Identifier 10.1109/52.2024
AbstractPlus | Full Text: PDF(848 KB) **IEEE JNL**
Rights and Permissions

2. **Program annotation in XML: a parse-tree based approach**
Power, J.F.; Malloy, B.A.;
Reverse Engineering, 2002. Proceedings. Ninth Working Conference on
29 Oct.-1 Nov. 2002 Page(s):190 - 198
Digital Object Identifier 10.1109/WCRE.2002.1173077
AbstractPlus | Full Text: PDF(274 KB) **IEEE CNF**
Rights and Permissions

3. **Proceedings of 2002 International Conference on Machine Learning and ((Cat.No.02EX583)**
Machine Learning and Cybernetics, 2002. Proceedings. 2002 International Cor
Volume 2, 4-5 Nov. 2002
Digital Object Identifier 10.1109/ICMLC.2002.1174376
AbstractPlus | Full Text: PDF(1409 KB) **IEEE CNF**
Rights and Permissions

Terms used
annotation program modify modification warning error

Found **23** of **2** searched out of **458**.

Sort results by: relevance

Display results: expanded form

Save results to a Binder

Search Tips

☐ Open results in a new window

Try an Advanced Search
Try this search in The Digital Library

Results 1 - 20 of 23 Result page: **1** 2 next

Relevance scale

1 Extended static checking for Java

Cormac Flanagan, K. Rustan M. Leino, Mark Lillibridge, Greg Nelson, James B. Saxe, Raymie Stata

May 2002 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2002 Conference on Programming language design and implementation PLDI '02**, Volume 37 Issue 5

Publisher: ACM Press

Full text available: pdf(257.73 KB) Additional Information: full citation, abstract, references, citings, index terms

Software development and maintenance are costly endeavors. The cost can be reduced if more software defects are detected earlier in the development cycle. This paper introduces the Extended Static Checker for Java (ESC/Java), an experimental compile-time program checker that finds common programming errors. The checker is powered by verification-condition generation and automatic theorem-proving techniques. It provides programmers with a simple annotation language with which programmer design de ...

Keywords: compile-time program checking

2 Types for safe locking: Static race detection for Java

Martin Abadi, Cormac Flanagan, Stephen N. Freund

March 2006 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 28 Issue 2

Publisher: ACM Press

Full text available: pdf(1.49 MB) Additional Information: full citation, abstract, references, index terms

This article presents a static race-detection analysis for multithreaded shared-memory programs, focusing on the Java programming language. The analysis is based on a type system that captures many common synchronization patterns. It supports classes with internal synchronization, classes that require client-side synchronization, and thread-local classes. In order to demonstrate the effectiveness of the type system, we have implemented it in a checker and applied it to over 40,000 lines of hand- ...

Keywords: Concurrent programs, race conditions, type inference, type system

3 Static detection of dynamic memory errors

David Evans

May 1996 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 1996 conference on Programming language design and implementation PLDI '96**, Volume 31 Issue 5

Publisher: ACM Press

Full text available: pdf(1.17 MB) Additional Information: full citation, abstract, references, citings, index terms

Many important classes of bugs result from invalid assumptions about the results of functions and the values of parameters and global variables. Using traditional methods, these bugs cannot be detected efficiently at compile-time, since detailed cross-procedural analyses would be required to determine the relevant assumptions. In this work, we introduce annotations to make certain assumptions explicit at interface points. An efficient static checking tool that exploits these annotations can dete ...

4 Dynamic program analysis: Invariant inference for static checking:

Jeremy W. Nimmer, Michael D. Ernst

November 2002 **Proceedings of the 10th ACM SIGSOFT symposium on Foundations of software engineering**

Publisher: ACM Press

Full text available: pdf(115.22 KB) Additional Information: full citation, abstract, references, citings, index terms

Static checking can verify the absence of errors in a program, but often requires written annotations or specifications. As a result, static checking can be difficult to use effectively: it can be difficult to determine a specification and tedious to annotate programs. Automated tools that aid the annotation process can decrease the cost of static checking and enable it to be more widely used.This paper describes an evaluation of the effectiveness of two techniques, one static and one dynamic, t ...

5 Session 2: dynamic program analysis: Invariant inference for static checking: an empirical evaluation

Jeremy W. Nimmer, Michael D. Ernst

November 2002 **ACM SIGSOFT Software Engineering Notes**, Volume 27 Issue 6

Publisher: ACM Press

Full text available: pdf(1.25 MB) Additional Information: full citation, abstract, references

Static checking can verify the absence of errors in a program, but often requires written annotations or specifications. As a result, static checking can be difficult to use effectively: it can be difficult to determine a specification and tedious to annotate programs. Automated tools that aid the annotation process can decrease the cost of static checking and enable it to be more widely used.This paper describes an evaluation of the effectiveness of two techniques, one static and one dynamic, t ...

6 A serializability violation detector for shared-memory server programs

Min Xu, Rastislav Bodík, Mark D. Hill

June 2005 **ACM SIGPLAN Notices , Proceedings of the 2005 ACM SIGPLAN conference on Programming language design and implementation PLDI '05**, Volume 40 Issue 6

Publisher: ACM Press

Full text available: pdf(326.11 KB) Additional Information: full citation, abstract, references, index terms

We aim to improve reliability of multithreaded programs by proposing a dynamic detector that detects potentially erroneous program executions and their causes. We design and evaluate a *Serializability Violation Detector* (SVD) that has two unique goals: (I) triggering automatic recovery from erroneous executions using backward error recovery (BER), or simply alerting users that a software error may have occurred; and (II) helping debug programs by revealing causes of error symptoms.Two pro ...

Keywords: multithreading, race conditions, serializability

7 Flow-sensitive type qualifiers

Jeffrey S. Foster, Tachio Terauchi, Alex Aiken

May 2002 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2002 Conference**

on Programming language design and implementation PLDI '02, Volume 37 Issue 5

Publisher: ACM Press

Full text available: pdf(298.28 KB) Additional Information: full citation, abstract, references, citings, index terms

We present a system for extending standard type systems with flow-sensitive type qualifiers. Users annotate their programs with type qualifiers, and inference checks that the annotations are correct. In our system only the type qualifiers are modeled flow-sensitively---the underlying standard types are unchanged, which allows us to obtain an efficient constraint-based inference algorithm that integrates flow-insensitive alias analysis, effect inference, and ideas from linear type systems to supp ...

Keywords: alias analysis, constraints, effect inference, flow-sensitivity, linux kernel, locking, restrict, type qualifiers, types

8 ARCHER: using symbolic, path-sensitive analysis to detect memory access errors

Yichen Xie, Andy Chou, Dawson Engler

September 2003 **ACM SIGSOFT Software Engineering Notes , Proceedings of the 9th European software engineering conference held jointly with 11th ACM SIGSOFT international symposium on Foundations of software engineering ESEC/FSE-11**, Volume 28 Issue 5

Publisher: ACM Press

Full text available: pdf(582.35 KB) Additional Information: full citation, abstract, references, citings, index terms

Memory corruption errors lead to non-deterministic, elusive crashes. This paper describes ARCHER (*ARray CHeckER*) a static, effective memory access checker. ARCHER uses path-sensitive, interprocedural symbolic analysis to bound the values of both variables and memory sizes. It evaluates known values using a constraint solver at every array access, pointer dereference, or call to a function that expects a size parameter. Accesses that violate constraints are flagged as errors. Those that ar ...

Keywords: buffer overflow, buffer overrun, error detection, memory access errors, security, static analysis

9 Detecting race conditions in large programs

Cormac Flanagan, Stephen N. Freund

June 2001 **Proceedings of the 2001 ACM SIGPLAN-SIGSOFT workshop on Program analysis for software tools and engineering**

Publisher: ACM Press

Full text available: pdf(163.40 KB) Additional Information: full citation, abstract, references, citings, index terms

The race condition checker \rcc{} statically identifies potential races in concurrent Java programs. This paper describes improvements to \rcc{} that enable it to be used on large, realistic programs. These improvements include not only extensions to the underlying analysis, but also an annotation inference algorithm and a user interface to help programmers understand warnings generated by the tool. Experience with programs containing up to 500,000 lines of code indicate that it is an ef ...

10 An instructional interpreter for basic

Avron Barr, Marian Beard

February 1976 **ACM SIGCSE Bulletin , ACM SIGCUE Outlook , Proceedings of the ACM SIGCSE-SIGCUE technical symposium on Computer science and education**, Volume 8 , 10 Issue 1 , SI

Publisher: ACM Press

Full text available: pdf(670.62 KB) Additional Information: full citation, abstract, references, citings, index terms

The BASIC Instructional Program (BIP) was developed to investigate tutorial modes of interaction in computer-assisted instruction (CAI). BIP is a problem-solving laboratory that helps students while they are solving introductory programming problems in the BASIC language. The problems are presented in an individualized sequence based on a representation of the structure of the curriculum and a model of the student's state of knowledge. This paper describes the BIP system, with emphasis on r ...

11 Implementation of the Sentry System

Sarah E. Chodrow, Mohamed G. Gouda

May 1994 Technical Report

Publisher: University of Texas at Austin

Additional Information: full citation, abstract

The sentry of a concurrent program P is a program that observes the execution of P, and issues a warning if P does not behave correctly with respect to a given set of logical properties (due to a programming error or a failure). The synchronization between the program and sentry is such that the program never waits for the sentry, the shared storage between them is very small (in fact linear in the number of program variables being observed), and the snapshots read by the sentry are consistent. ...

12 Grammatical analysis by computer of the Lancaster-Oslo/Bergen (LOB) corpus of British English texts

Andrew David Beale

July 1985 **Proceedings of the 23rd annual meeting on Association for Computational Linguistics**

Publisher: Association for Computational Linguistics

Full text available: pdf(581.37 KB) Publisher Site

Additional Information: full citation, abstract, references, citings

Research has been under way at the unit for Computer Research on the English Language at the University of Lancaster, England, to develop a suite of computer programs which provide a detailed grammatical analysis of the LOB corpus, a collection of about 1 million words of British English texts available in machine readable form.The first phrase of the project, completed in September 1983, produced a grammatically annotated version of the corpus giving a tag showing the word class of each word to ...

13 A type and effect system for atomicity

Cormac Flanagan, Shaz Qadeer

May 2003 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2003 conference on Programming language design and implementation PLDI '03**, Volume 38 Issue 5

Publisher: ACM Press

Full text available: pdf(266.52 KB)

Additional Information: full citation, abstract, references, citings, index terms

Ensuring the correctness of multithreaded programs is difficult, due to the potential for unexpected and nondeterministic interactions between threads. Previous work addressed this problem by devising tools for detecting *race conditions*, a situation where two threads simultaneously access the same data variable, and at least one of the accesses is a write. However, verifying the absence of such simultaneous-access race conditions is neither necessary nor sufficient to ensure the absence o ...

Keywords: atomicity, multithreading, race conditions, static checking

14 A probabilistic approach to grammatical analysis of written English by computer

Andrew David Beale

March 1985 **Proceedings of the second conference on European chapter of the Association for Computational Linguistics**

Publisher: Association for Computational Linguistics

Full text available: pdf(700.26 KB) Additional Information: full citation, abstract, references, citings
Publisher Site

Work at the Unit for Computer Research on the English Language at the University of Lancaster has been directed towards producing a grammatically annotated version of the Lancaster-Oslo/Bergen (LOB) Corpus of written British English texts as the preliminary stage in developing computer programs and data files for providing a grammatical analysis of unrestricted English text.From 1981--83, a suite of PASCAL programs was devised to automatically produce a single level of grammatical description wi ...

15 Automatic generation of program specifications

Jeremy W. Nimmer, Michael D. Ernst

July 2002 **ACM SIGSOFT Software Engineering Notes , Proceedings of the 2002 ACM SIGSOFT international symposium on Software testing and analysis ISSTA '02**, Volume 27 Issue 4

Publisher: ACM Press

Full text available: pdf(154.41 KB) Additional Information: full citation, abstract, references, citings

Producing specifications by dynamic (runtime) analysis of program executions is potentially unsound, because the analyzed executions may not fully characterize all possible executions of the program. In practice, how accurate are the results of a dynamic analysis? This paper describes the results of an investigation into this question, determining how much specifications generalized from program runs must be changed in order to be verified by a static checker. Surprisingly, small test suites cap ...

16 Type-based race detection for Java

Cormac Flanagan, Stephen N. Freund

May 2000 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2000 conference on Programming language design and implementation PLDI '00**, Volume 35 Issue 5

Publisher: ACM Press

Full text available: pdf(237.37 KB) Additional Information: full citation, abstract, references, citings, index terms

This paper presents a static race detection analysis for multithreaded Java programs. Our analysis is based on a formal type system that is capable of capturing many common synchronization patterns. These patterns include classes with internal synchronization, classes thatrequire client-side synchronization, and thread-local classes. Experience checking over 40,000 lines of Java code with the type system demonstrates that it is an effective approach for eliminating races conditions. On lar ...

17 Untrusted hosts and confidentiality: secure program partitioning

Steve Zdancewic, Lantian Zheng, Nathaniel Nystrom, Andrew C. Myers

October 2001 **ACM SIGOPS Operating Systems Review , Proceedings of the eighteenth ACM symposium on Operating systems principles SOSP '01**, Volume 35 Issue 5

Publisher: ACM Press

Full text available: pdf(1.36 MB) Additional Information: full citation, abstract, references, citings, index terms

This paper presents secure program partitioning, a language-based technique for protecting confidential data during computation in distributed systems containing mutually untrusted hosts. Confidentiality and integrity policies can be expressed by annotating programs with security types that constrain information flow; these programs can then be partitioned automatically to run securely on heterogeneously trusted hosts. The resulting communicating subprograms collectively implement the original p ...

18 Secure program partitioning

Steve Zdancewic, Lantian Zheng, Nathaniel Nystrom, Andrew C. Myers

August 2002 **ACM Transactions on Computer Systems (TOCS)**, Volume 20 Issue 3

Publisher: ACM Press

Full text available: pdf(497.12 KB) Additional Information: full citation, abstract, references, index terms

This paper presents secure program partitioning, a language-based technique for protecting confidential data during computation in distributed systems containing mutually untrusted hosts. Confidentiality and integrity policies can be expressed by annotating programs with security types that constrain information flow; these programs can then be partitioned automatically to run securely on heterogeneously trusted hosts. The resulting communicating subprograms collectively implement the original p ...

Keywords: Confidentiality, declassification, distributed systems, downgrading, integrity, mutual distrust, secrecy, security policies, type systems

19 An experience with a Prolog-based object-oriented language

Koichi Fukunaga, Shin-ichi Hirose

June 1986 **ACM SIGPLAN Notices , Conference proceedings on Object-oriented programming systems, languages and applications OOPLSA '86**, Volume 21 Issue 11

Publisher: ACM Press

Full text available: pdf(793.08 KB) Additional Information: full citation, abstract, references, citings, index terms

This paper presents an experience with a programming language SPOOL which is based on the combination of object-oriented programming and logic programming. This language inherits the capability of knowledge base organization from object-oriented programming and its expressive power from logic programming.The experience of the application of SPOOL to the program annotation system showed that this combination was quite useful to formalize domain knowledge into declarative data type ...

20 Security and privacy: Securing web application code by static analysis and runtime protection

Yao-Wen Huang, Fang Yu, Christian Hang, Chung-Hung Tsai, Der-Tsai Lee, Sy-Yen Kuo

May 2004 **Proceedings of the 13th international conference on World Wide Web**

Publisher: ACM Press

Full text available: pdf(2.67 MB) Additional Information: full citation, abstract, references, index terms

Security remains a major roadblock to universal acceptance of the Web for many kinds of transactions, especially since the recent sharp increase in remotely exploitable vulnerabilities have been attributed to Web application bugs. Many verification tools are discovering previously unknown vulnerabilities in legacy C programs, raising hopes that the same success can be achieved with Web applications. In this paper, we describe a sound and holistic approach to ensuring Web application security. Vi ...

Keywords: information flow, noninterference, program security, security vulnerabilities, type systems, verification, web application security

Terms used **annotation program warning modify modification**

Found **1** of **176,279**

Sort results by: relevance

Display results: expanded form

Save results to a Binder

Search Tips

Open results in a new window

Try an Advanced Search

Try this search in The ACM Guide

Results 1 - 1 of 1

Relevance scale

1 Extended static checking for Java

Cormac Flanagan, K. Rustan M. Leino, Mark Lillibridge, Greg Nelson, James B. Saxe, Raymie Stata

May 2002 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2002 Conference on Programming language design and implementation PLDI '02**, Volume 37 Issue 5

Publisher: ACM Press

Full text available: pdf(257.73 KB) Additional Information: full citation, abstract, references, citings, index terms

Software development and maintenance are costly endeavors. The cost can be reduced if more software defects are detected earlier in the development cycle. This paper introduces the Extended Static Checker for Java (ESC/Java), an experimental compile-time program checker that finds common programming errors. The checker is powered by verification-condition generation and automatic theorem-proving techniques. It provides programmers with a simple annotation language with which programmer design de ...

Keywords: compile-time program checking

Results 1 - 1 of 1